On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis

Jose Camacho-Collados
School of Computer Science
and Informatics
Cardiff University
camachocolladosj@cardiff.ac.uk

Mohammad Taher Pilehvar
School of Computer Engineering
Iran University of
Science and Technology
pilehvar@iust.ac.ir

Abstract

Text preprocessing is often the first step in the pipeline of a Natural Language Processing (NLP) system, with potential impact in its final performance. Despite its importance, text preprocessing has not received much attention in the deep learning literature. In this paper we investigate the impact of simple text preprocessing decisions (particularly tokenizing, lemmatizing, lowercasing and multiword grouping) on the performance of a standard neural text classifier. We perform an extensive evaluation on standard benchmarks from text categorization and sentiment analysis. While our experiments show that a simple tokenization of input text is generally adequate, they also highlight significant degrees of variability across preprocessing techniques. This reveals the importance of paying attention to this usually-overlooked step in the pipeline, particularly when comparing different models. Finally, our evaluation provides insights into the best preprocessing practices for training word embeddings.

1 Introduction

Words are often considered as the basic constituents of texts for many languages, including English.[1] The first module in an NLP pipeline is a tokenizer which transforms texts to sequences of words. However, in practise, other preprocessing techniques can be (and are) further used together with tokenization. These include lemmatization, lowercasing and multiword grouping, among others. Although these preprocessing decisions have been studied in the context of conventional text classification techniques (Leopold and Kindermann, 2002; Uysal and Gunal, 2014), little attention has been paid to them in the more recent neural-based models. The most similar study to ours is Zhang and LeCun (2017), which analyzed different encoding levels for English and Asian languages such as Chinese, Japanese and Korean. As opposed to our work, their analysis was focused on UTF-8 bytes, characters, words, romanized characters and romanized words as encoding levels, rather than the preprocessing techniques analyzed in this paper.

Additionally, word embeddings have been shown to play an important role in boosting the generalization capabilities of neural systems (Goldberg, 2016; Camacho-Collados and Pilehvar, 2018). However, while some studies have focused on intrinsically analyzing the role of lemmatization in their underlying training corpus (Ebert et al., 2016; Kuznetsov and Gurevych, 2018), the impact on their extrinsic performance when integrated into a neural network architecture has remained understudied.[2]

In this paper we focus on the role of preprocessing the input text, particularly in how it is split into individual (meaning-bearing) tokens and how it affects the performance of standard neural text classification models based on Convolutional Neural Networks (LeCun et al., 2010; Kim, 2014, CNN). CNNs have proven to be effective in a wide range of NLP applications, in-

[1] Note that although word-based models are mainstream in NLP in general and text classification in particular, recent work has also considered other linguistic units, such as characters (Kim et al., 2016; Xiao and Cho, 2016) or word senses (Li and Jurafsky, 2015; Flekova and Gurevych, 2016; Pilehvar et al., 2017). These techniques require a different kind of preprocessing and, while they have been shown effective in various settings, in this work we only focus on the mainstream word-based models.

[2] Not only the preprocessing of the corpus may play an important role but also its nature, domain, etc. Levy et al. (2015) also showed how small hyperparameter variations may have an impact on the performance of word embeddings. However, these considerations remain out of the scope of this paper.

cluding text classification tasks such as topic categorization (Johnson and Zhang, 2015; Tang et al., 2015; Xiao and Cho, 2016; Conneau et al., 2017) and polarity detection (Kalchbrenner et al., 2014; Kim, 2014; Dos Santos and Gatti, 2014; Yin et al., 2017), which are the tasks considered in this work. The goal of our evaluation study is to find answers to the following two questions:

1. Are neural network architectures (in particular CNNs) affected by seemingly small preprocessing decisions in the input text?

2. Does the preprocessing of the embeddings' underlying training corpus have an impact on the final performance of a state-of-the-art neural network text classifier?

According to our experiments in topic categorization and polarity detection, these decisions are important in certain cases. Moreover, we shed some light on the motivations of each preprocessing decision and provide some hints on how to normalize the input corpus to better suit each setting.

The accompanying materials of this submission can be downloaded at the following repository: github.com/pedrada88/preproc-textclassification.

2 Text Preprocessing

Given an input text, words are gathered as input units of classification models through tokenization. We refer to the corpus which is only tokenized as ***vanilla***. For example, given the sentence "Apple is asking its manufacturers to move MacBook Air production to the United States." (running example), the vanilla tokenized text would be as follows (white spaces delimiting different word units):

Apple is asking its manufacturers to move MacBook Air production to the United States .

We additionally consider three simple preprocessing techniques to be applied to an input text: lowercasing (Section 2.1), lemmatizing (Section 2.2) and multiword grouping (Section 2.3).

2.1 Lowercasing

This is the simplest preprocessing technique which consists of lowercasing each single token of the input text:

apple is asking its manufacturers to move macbook air production to the united states .

Due to its simplicity, lowercasing has been a popular practice in modules of deep learning libraries and word embedding packages (Pennington et al., 2014; Faruqui et al., 2015). Despite its desirable property of reducing sparsity and vocabulary size, lowercasing may negatively impact system's performance by increasing ambiguity. For instance, the *Apple* company in our example and the *apple* fruit would be considered as identical entities.

2.2 Lemmatizing

The process of lemmatizing consists of replacing a given token with its corresponding lemma:

Apple be ask its manufacturer to move MacBook Air production to the United States .

Lemmatization has been traditionally a standard preprocessing technique for linear text classification systems (Mullen and Collier, 2004; Toman et al., 2006; Hassan et al., 2007). However, it is rarely used as a preprocessing stage in neural-based systems. The main idea behind lemmatization is to reduce sparsity, as different inflected forms of the same lemma may occur infrequently (or not at all) during training. However, this may come at the cost of neglecting important syntactic nuances.

2.3 Multiword grouping

This last preprocessing technique consists of grouping consecutive tokens together into a single token if found in a given inventory:

Apple is asking its manufacturers to move MacBook_Air production to the United_States .

The motivation behind this step lies in the idiosyncratic nature of multiword expressions (Sag et al., 2002), e.g. *United States* in the example. The meaning of these multiword expressions are often hardly traceable from their individual tokens. As a result, treating multiwords as single units may lead to better training of a given model. Because of this, word embedding toolkits such as Word2vec propose statistical approaches for extracting these multiwords, or directly include multiwords along with single words in their pretrained embedding spaces (Mikolov et al., 2013b).

3 Evaluation

We considered two tasks for our experiments: **topic categorization**, i.e. assigning a topic to a given document from a pre-defined set of topics, and **polarity detection**, i.e. detecting if the sentiment of a given piece of text is positive or negative (Dong et al., 2015). Two different settings were studied: (1) word embedding's training corpus and the evaluation dataset were preprocessed in a similar manner (Section 3.2); and (2) the two were preprocessed differently (Section 3.3). In what follows we describe the common experimental setting as well as the datasets and preprocessing used for the evaluation.

3.1 Experimental setup

We tried with two classification models. The first one is a standard CNN model similar to that of Kim (2014), using ReLU (Nair and Hinton, 2010) as non-linear activation function. In the second model, we add a recurrent layer (specifically an LSTM (Hochreiter and Schmidhuber, 1997)) before passing the pooled features directly to the fully connected softmax layer.[3] The inclusion of this LSTM layer has been shown to be able to effectively replace multiple layers of convolution and be beneficial particularly for large inputs (Xiao and Cho, 2016). These models were used for both topic categorization and polarity detection tasks, with slight hyperparameter variations given their different natures (mainly in their text size) which were fixed across all datasets. The embedding layer was initialized using 300-dimensional CBOW Word2vec embeddings (Mikolov et al., 2013a) trained on the 3B-word UMBC WebBase corpus (Han et al., 2013) with standard hyperparameters[4].

Evaluation datasets. For the topic categorization task we used the **BBC** news dataset[5] (Greene and Cunningham, 2006), **20News** (Lang, 1995), **Reuters**[6] (Lewis et al., 2004) and **Ohsumed**[7]. **PL04** (Pang and Lee, 2004), **PL05**[8] (Pang and Lee, 2005), **RTC**[9], **IMDB** (Maas et al., 2011) and the Stanford sentiment dataset[10] (Socher et al., 2013, **SF**) were considered for polarity detection. Statistics of the versions of the datasets used are displayed in Table 1.[11] For both tasks the evaluation was carried out either by 10-fold cross-validation or using the train-test splits of the datasets, in case of availability.

	Dataset	Type	Labels	# of docs	Eval.
TOPIC	BBC	News	5	2,225	10-cross
	20News	News	6	18,846	Train-test
	Reuters	News	8	9,178	10-cross
	Ohsumed	Medical	23	23,166	Train-test
POLARITY	RTC	Snippets	2	438,000	Train-test
	IMDB	Reviews	2	50,000	Train-test
	PL05	Snippets	2	10,662	10-cross
	PL04	Reviews	2	2,000	10-cross
	Stanford	Phrases	2	119,783	10-cross

Table 1: Evaluation datasets for topic categorization and polarity detection.

Preprocessing. Four different techniques (see Section 2) were used to preprocess the datasets as well as the corpus which was used to train word embeddings (i.e. UMBC). For tokenization and lemmatization we relied on Stanford CoreNLP (Manning et al., 2014). As for multiwords, we used the phrases from the pre-trained Google News Word2vec vectors, which were obtained using a simple statistical approach (Mikolov et al., 2013b).[12]

3.2 Experiment 1: Preprocessing effect

Table 2 shows the accuracy[13] of the classification models using our four preprocessing techniques. We observe a certain variability of results depending on the preprocessing techniques used (aver-

[3] The code for this CNN implementation is the same as in (Pilehvar et al., 2017), which is available at https://github.com/pilehvar/sensecnn

[4] Context window of 5 words and hierarchical softmax.

[5] http://mlg.ucd.ie/datasets/bbc.html

[6] Due to the large number of labels in the original Reuters (i.e. 91) and to be consistent with the other datasets, we reduce the dataset to its 8 most frequent labels, a reduction already performed in previous works (Sebastiani, 2002).

[7] ftp://medir.ohsu.edu/pub/ohsumed

[8] Both PL04 and PL05 were downloaded from http://www.cs.cornell.edu/people/pabo/movie-review-data/

[9] http://www.rottentomatoes.com

[10] We mapped the numerical value of phrases to either negative (from 0 to 0.4) or positive (from 0.6 to 1), removing the neutral phrases according to the scale (from 0.4 to 0.6).

[11] For the datasets with train-test partitions, the sizes of the test sets are the following: 7,532 for 20News; 12,733 for Ohsumed; 25,000 for IMDb; and 1,000 for RTC.

[12] For future work it would be interesting to explore more complex methods to learn embeddings for multiword expressions (Yin and Schütze, 2014; Poliak et al., 2017).

[13] Computed by averaging accuracy of two different runs. The statistical significance was calculated according to an unpaired t-test at the 5% significance level.

		Topic categorization				Polarity detection				
	Preprocessing	**BBC**	**20News**	**Reuters**	**Ohsumed**	**RTC**	**IMDB**	**PL05**	**PL04**	**SF**
CNN	Vanilla	94.6	89.2	93.7	35.3	**83.2**	87.5	76.3	58.7†	**91.2**
	Lowercased	94.8	**89.8**	**94.2**	**36.0**	83.0	84.2†	76.1	59.6†	91.1
	Lemmatized	95.4	89.4	94.0	35.9	83.1	86.8†	75.8†	**64.2**	**91.2**
	Multiword	**95.5**	89.6	93.4†	34.3†	**83.2**	**87.9**	**77.0**	59.1†	**91.2**
CNN+LSTM	Vanilla	**97.0**	90.7	93.1	30.8†	**84.8**	**88.9**	79.1	71.4	87.1
	Lowercased	96.4	**90.9**	93.0	**37.5**	84.0	88.3†	**79.5**	**73.3**	87.1
	Lemmatized	95.8†	90.5	**93.2**	37.1	84.4	87.7†	78.7	72.6	86.8†
	Multiword	96.2	89.8†	92.7†	29.0†	84.0	**88.9**	79.2	67.0†	**87.3**

Table 2: Accuracy on the topic categorization and polarity detection tasks using various preprocessing techniques for the CNN and CNN+LSTM models. † indicates results that are statistically significant with respect to the top result.

age variability[14] of $\pm 2.4\%$ for the CNN+LSTM model, including a statistical significance gap in seven of the nine datasets), which proves the influence of preprocessing on the final results. It is perhaps not surprising that the lowest variance of results is seen in the datasets with the larger training data (i.e. RTC and Stanford). This suggests that the preprocessing decisions are not so important when the training data is large enough, but they are indeed relevant in benchmarks where the training data is limited.

As far as the individual preprocessing techniques are concerned, the vanilla setting (tokenization only) proves to be consistent across datasets and tasks, as it performs in the same ballpark as the best result in 8 of the 9 datasets for both models (with no noticeable differences between topic categorization and polarity detection). The only topic categorization dataset in which tokenization does not seem enough is Ohsumed, which, unlike the more general nature of other categorization datasets (news), belongs to a specialized domain (medical) for which fine-grained distinctions are required to classify cardiovascular diseases. In particular for this dataset, word embeddings trained on a general-domain corpus like UMBC may not accurately capture the specialized meaning of medical terms and hence, sparsity becomes an issue. In fact, lowercasing and lemmatizing, which are mainly aimed at reducing sparsity, outperform the vanilla setting by over six points in the CNN+LSTM setting and clearly outperform the other preprocessing techniques on the single CNN model as well.

Nevertheless, the use of more complex preprocessing techniques such as lemmatization and multiword grouping does not help in general. Even though lemmatization has proved useful in conventional linear models as an effective way to deal with sparsity (Mullen and Collier, 2004; Toman et al., 2006), neural network architectures seem to be more capable of overcoming sparsity thanks to the generalization power of word embeddings.

3.3 Experiment 2: Cross-preprocessing

This experiment aims at studying the impact of using different word embeddings (with differently preprocessed training corpora) on tokenized datasets (vanilla setting). Table 3 shows the results for this experiment. In this experiment we observe a different trend, with multiword-enhanced vectors exhibiting a better performance both on the single CNN model (best overall performance in seven of the nine datasets) and on the CNN+LSTM model (best performance in four datasets and in the same ballpark as the best results in four of the remaining five datasets). In this case the same set of words is learnt but single tokens inside multiword expressions are not trained. Instead, these single tokens are considered in isolation only, without the added *noise* when considered inside the multiword expression as well. For instance, the word *Apple* has a clearly different meaning in isolation from the one inside

[14] Average variability was the result of averaging the variability of each dataset, which was computed as the difference between the best and the worst preprocessing performances.

	Embedding Preprocessing	Topic categorization				Polarity detection				
		BBC	**20News**	**Reuters**	**Ohsumed**	**RTC**	**IMDB**	**PL05**	**PL04**	**SF**
CNN	Vanilla	94.6	89.2	93.7	35.3	83.2	87.5†	**76.3**	58.7†	**91.2**
	Lowercased	93.9†	84.6†	**93.9**	**36.2**	83.2	85.4†	**76.3**	60.0†	91.1
	Lemmatized	94.5	88.7†	93.8	35.4	83.0	86.8†	75.6	62.5	**91.2**
	Multiword	**95.6**	**89.7**	**93.9**	35.2	**83.3**	**88.1**	75.9	**63.1**	**91.2**
CNN+LSTM	Vanilla	97.0	90.7†	**93.1**	30.8†	**84.8**	**88.9**	79.1	71.4	87.1†
	Lowercased	96.4	**91.8**	92.5†	30.2†	84.5	88.0†	79.0	**74.2**	87.4
	Lemmatized	96.6	91.5	92.5†	31.7†	83.9	86.6†	78.4†	67.7†	87.3
	Multiword	**97.3**	91.3	92.8	**33.6**	84.3	87.3†	**79.5**	71.8	**87.5**

Table 3: Cross-preprocessing evaluation: accuracy on the topic categorization and polarity detection tasks using different sets of word embeddings to initialize the embedding layer of the two classifiers. All datasets were preprocessed similarly according to the vanilla setting. † indicates results that are statistically significant with respect to the top result.

the multiword expression *Big_Apple*, hence it can be seen as beneficial not to train the word *Apple* when part of this multiword expression. Interestingly, using multiword-wise embeddings on the vanilla setting leads to consistently better results than using them on the same multiword-grouped preprocessed dataset in eight of the nine datasets. This could provide hints on the excellent results provided by pre-trained Word2vec embeddings trained on the Google News corpus, which learns multiwords similarly to our setting.

Apart from this somewhat surprising finding, the use of the embeddings trained on a simple tokenized corpus (i.e. vanilla) proved again competitive, as different preprocessing techniques such as lowercasing and lemmatizing do not seem to help. In fact, the relatively weaker performance of lemmatization and lowercasing in this cross-processing experiment is somehow expected as the coverage of word embeddings in vanilla-tokenized datasets is limited, e.g., many entities which are capitalized in the datasets are not covered in the case of lowercasing, and inflected forms are missing in the case of lemmatizing.

4 Conclusions

In this paper we analyzed the impact of simple text preprocessing decisions on the performance of a standard word-based neural text classifier. Our evaluations highlight the importance of being careful in the choice of how to preprocess our data and to be consistent when comparing different systems. In general, a simple tokenization works equally or better than more complex preprocessing techniques such as lemmatization or multiword grouping, except for domain-specific datasets (such as the medical dataset in our experiments) in which sole tokenization performs poorly. Additionally, word embeddings trained on multiword-grouped corpora perform surprisingly well when applied to simple tokenized datasets. This property has often been overlooked and, to the best of our knowledge, we test the hypothesis for the first time. In fact, this finding could partially explain the long-lasting success of pre-trained Word2vec embeddings, which specifically learn multiword embeddings as part of their pipeline (Mikolov et al., 2013b).

Moreover, our analysis shows that there is a high variance in the results depending on the preprocessing choice ($\pm 2.4\%$ on average for the best performing model), especially when the training data is not large enough to generalize. Further analysis and experimentation would be required to fully understand the significance of these results; but, this work can be viewed as a starting point for studying the impact of text preprocessing in deep learning models. We hope that our findings will encourage future researchers to carefully select and report these preprocessing decisions when evaluating or comparing different models. Finally, as future work, we plan to extend our analysis to other tasks (e.g. question answering), languages (particularly morphologically rich languages for which these results may vary) and preprocessing techniques (e.g. stopword removal or part-of-speech tagging).

Acknowledgments

Jose Camacho-Collados is supported by the ERC Starting Grant 637277.

References

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research (JAIR)*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of EACL*, pages 1107–1116, Valencia, Spain.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*, pages 69–78.

Sebastian Ebert, Thomas Müller, and Hinrich Schütze. 2016. Lamb: A good shepherd of morphologically rich languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 742–752.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, pages 1606–1615.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *Proceedings of ACL*, Berlin, Germany.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International conference on Machine learning*, pages 377–384. ACM.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52.

Samer Hassan, Rada Mihalcea, and Carmen Banea. 2007. Random walk term weighting for improved text classification. *International Journal of Semantic Computing*, 1(04):421–439.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of NAACL*, pages 103–112, Denver, Colorado.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*.

Ilia Kuznetsov and Iryna Gurevych. 2018. From text to lexicon: Bridging the gap between word embeddings and lexical resources. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 233–244.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423–444.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

David D. Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of EMNLP*, Lisbon, Portugal.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150, Portland, Oregon, USA.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proceedings of ACL*, Vancouver, Canada.

Adam Poliak, Pushpendre Rastogi, M. Patrick Martin, and Benjamin Van Durme. 2017. Efficient, compositional, order-sensitive n-gram embeddings. In *Proceedings of EACL*.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing With Compositional Vector Grammars. In *Proceedings of EMNLP*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Michal Toman, Roman Tesar, and Karel Jezek. 2006. Influence of word normalization on text classification. *Proceedings of InSciT*, 4:354–358.

Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR*, abs/1602.00367.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Wenpeng Yin and Hinrich Schütze. 2014. An exploration of embeddings for generalized phrases. In *ACL (Student Research Workshop)*, pages 41–47.

Xiang Zhang and Yann LeCun. 2017. Which encoding is the best for text classification in chinese, english, japanese and korean? *arXiv preprint arXiv:1708.02657*.